Universitat de Lleida

TREBALL FINAL DE GRAU

ESCOLA POLITÈCNICA SUPERIOR
UNIVERSITAT DE LLEIDA
INSPIRING THE FUTURE

Estudiant: **Mireia Moix Atienza**

Titulació: **Doble grau en Enginyeria Informàtica i ADE**

Títol de Treball Final de Grau: **eFat, a prototype of a Cloud decision support system for optimal delivery of fattened pigs to the abattoir**

Director/a: **Lluís Miquel Pla Aragones, Jordi Mateo Fornés**

Presentació:

Mes: Juny

Any: 2019

Table of Contents

Table of Figures

Table of Tables

ABSTRACT

This project consists in developing a decision support system and its integration in a cloud platform to offer optimization models as a service integrated in a sole application accessible from different devices. It has two parts due to the double degree in Computer Engineering and Administration and Business Management requirement.

The part related to the Administration and Business Management degree is based on a mixed integer linear-programming (MILP) model to support marketing decisions on fattening farms without individual weight control. It includes the development of the running program to solve the MILP model and the assessment for specific factors. The objective is to determine the optimal delivery time and quality bonus of the fattened pigs to the slaughterhouse and thus, getting the best value for the producer.

This mathematical model maximizes the revenue and reduces the risk of neglecting opportunity costs by taking into account important key factors such as: homogeneity of the batch, price-grid system, transportation cost, among others. The analysis performed on this study demonstrated the importance of these different factors that affect significantly on the optimal marketing policy of fattening farms, and also in a vertically integrated company.

The second part is related to the Computer Engineering degree and concerns the development of a cloud decision support system (cDSS). The cDSS takes form of a client application that allows users to run the mathematical model introduced previously and through it, depict their best marketing decisions for a pig farmer. Through this application, the farmer can store data and update information of the farms, make requests from different devices or visualize an output display of the results obtained from the model.

1. INTRODUCTION

Many real-world decision-making situations arise from agribusiness. This sector is becoming more and more complex, technical and competitive. Although in this field significant work has being done, there is still much to do into the agribusiness area. In addiction, pig sector plays a vital role in many economies around the world. The complexity of mathematical models is an issue for their practical use and its implementation requires a previous understanding of the real problem, formulate, model, test, receive feedback, make corrections and implement the solution.

Pig sector is experiencing expansion and continuous growth in recent years. With regards to Spain, during the year 2017, it has reached a turnover of 15,000 million euros, largely due to the increase in sales abroad, standing at around 13% and reaching 5,000 million euros (in 2008 were around 2,400 million). Thus, Spain is the third power in terms of world trade in pork products, being present in more than 130 countries [1]. Furthermore, in Spain, the pig sector represents one third part of the livestock gross product while in Catalonia, one region of Spain, represents one-half of the livestock gross sector. Because of that, the pig sector is a valued sector in the Catalan agriculture.

The fattening period is one of the most important stages in pig production because it is at the end of the pig farming process. A pig company can see whether they have profit or not at this moment after the whole life cycle of pigs. In this period a batch of pigs are sent to the slaughterhouse, so each decision made about the delivery of fattened pigs from the fattening farm to the slaughterhouse may lead the company to succeed or to bankrupt.

Every pig producer wants to maximize their income over time and to reduce their risks of losses and that would be one of the main objectives of a business plan. However, determining the optimal delivery time and quality of the fattened pigs to the slaughterhouse is not an easy task to do. This fact is what the fattening farms have to face with little information though it is usually the most pressing decision-making problem. Thus, the combination of decision support systems embedding mathematical models with cloud systems allows overcoming some of the problems related to high performance computation, availability of useful results and control of production. The producer will be also able to react faster to unforeseen problems, re-planning and getting better knowledge about the weakest parts of the system that they have to improve.

For this project the Cloud platform plays a big role as well as the mathematical model, which is actually the most important part. Farmers will be able to control their farm more tightly through a friendly interface in which also enable them to run the model and help them to make decisions by seeing the results of their farm. Therefore, this cloud platform offers optimization models as a service integrated in a sole application accessible from different devices.

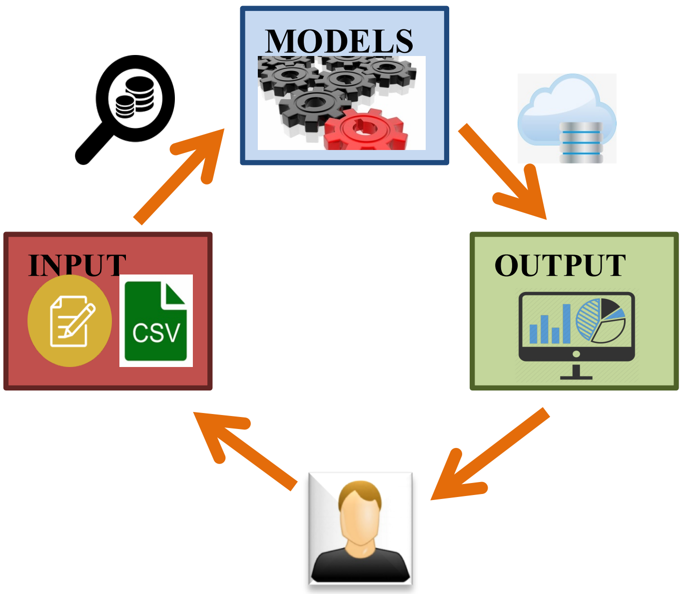


Figure 1: Concept of the cloud platform

Source 1: Own

Technically, the application developed in this project allows users to run the mixed integer linear-programming (MILP) model and through it, depict their best marketing decisions for pig grower, who operates under an AIAO strategy, can take. Moreover, through this application the farmer is able to store data and information from their farms in the database and can make requests from different devices they want. This request will end up with an output display, in other words, with some visual results with the information that the user has requested.

2. METHODOLOGY

2.1. MATHEMATICAL MODEL

2.1.1. TECHNOLOGIES INVOLVED

Initially, the mathematical model embedded in the cloud system was formulated with the modelling language OPL and and data stored and retrieved in MS Excel. This modelling language OPL (**O**ptimization **P**rogramming **L**anguage) is compatible with the CPLEX software package (IBM ILOG CPLEX). The solver used to solve the linear model instance is CPLEX version 12.8. Additionally, CPLEX provides functions to be launched from C++, Java and Python languages and also enables the developer to use Microsoft Excel to retrieve data parameters and store results.

Making use of these mentioned technologies implied that in order to use this model a license of OPL and Microsoft Excel was needed. Consequently, in this process, several neglected points emerged, since it was a rudimentary process, complex and difficult to maintain out of academic purposes. For this reason, this project proposes to substitute these licensed technologies. This replacement is based on free software tools, automating and making transparent the user's use. Therefore, the Pyomo has been used.

Figure 2: Logo of IBM CPLEX, Pyomo and PyCharm
Source 2: Official Websites

Pyomo is the modelling language used for this project, which is an open-source and based on Python. Pyomo means **Py**thon **O**ptimization **M**odelling **O**bjects and it is used to solve complex real-world applications. This tool allows the researchers to use different solvers and input sources, easiness to model, access to the full power of a solver and to a broad range of tools, offers helpful modelling extensions and solver-independent models.

Having a look at the *Figure 3*, the complexity that Pyomo supports can be seen and also the previous mentioned advantages that this tool offers.

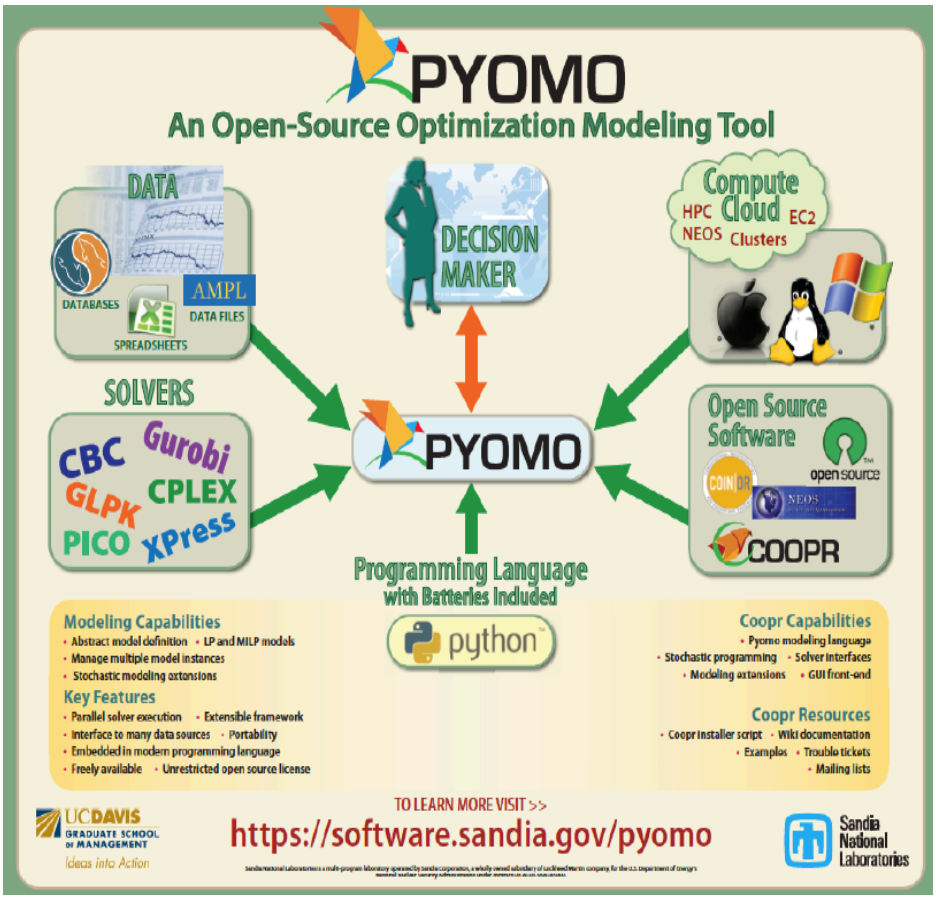


Figure 3: Pyomo operating diagram
Source 3: Pyomo website

In this project, in regards to data exchange, different types of files have been used: a configuration file, a CSV (**C**omma-**S**eparated **V**alues) file, which uses commas in order to separate field values, and a JSON (**J**ava**S**cript **O**bject **N**otation) file, which consist in attribute–value pairs and they are easy to be treated. On the other hand, CPLEX is the solver that has been used and Stormy server as a private cloud system. Finally, Python has been used as the programming language.

2.1.2. OPERATION OF FATTENING FARMS

The life cycle of pig goes through different stages in which the fattening period is considered one of the most important ones since the income is produced after selling them to the slaughterhouse. This project is based on the last stage; fattening. However, in order to understand deeply the meaning of the model, better to look back to the previous stages to have a follow-up. The stages of the production cycle are the following ones: the sow farm, the rearing farm and the fattening farm. After this cycle, the pig is sent to the slaughterhouse when they seem to reach the market weight.
The first stage is focus on maximizing the birth of piglets. The second stage, the rearing farm, the weaned piglets are prepared for the last stage, which is at the fattening farm.

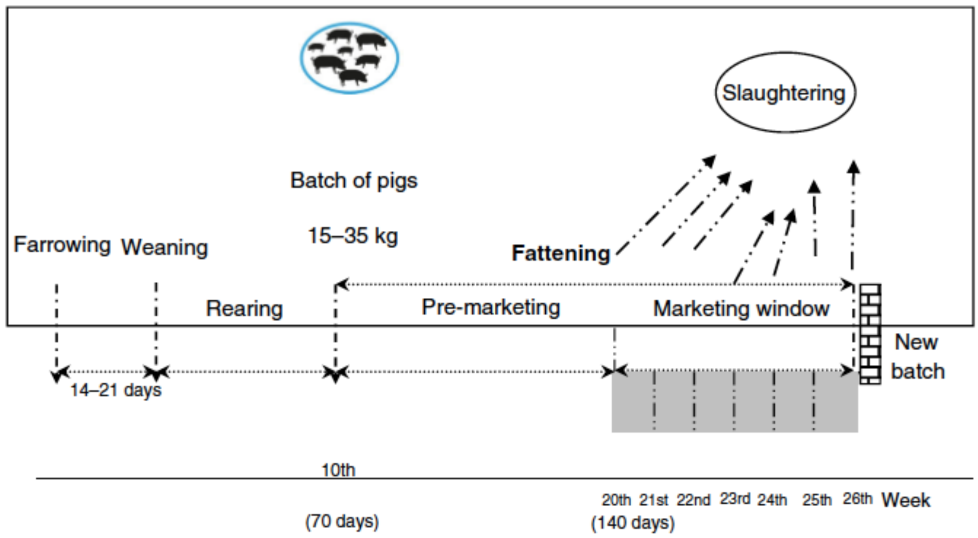


Figure 4: The life cycle of pigs during the fattening process
Source 4: CSIRO PUBLISHING - Insights to optimise marketing decisions on pig-grower farms

The fattening farm, as its name indicates, is the stage where the pig will be in its process of growing. This is a very uncertain process since the farmer can not have everything under control everything. There are different factors that may have a enormous impact on economic activity and thus, on the optimal marketing policy of fattening. This factors are the duration of the fattening period, the grading price system, transportation cost, when and how many pigs to deliver to the slaughterhouse and weight homogeneity of the batch.

The fattening farm’s management policy is All-In-All-Out (AIAO), which consist on selling all the batch of pigs before a new one is brought onto the farm. Once the last pig is sent to the slaughterhouse, cleaning and sanitized process must be done before the arrival of the new batch of pigs.

Deciding the exact time that pigs have to be sent to the slaughterhouse is one of the most critical and complex decisions in the production cycle. Here is when you want to get the best economic rewards and avoid to be penalized by the slaughterhouse. Therefore, on this decision-making process is where the model can play a fundamental role.

The model’s purpose in the pig sector organized into vertically integrated pork supply-chain (PSC) is to maximize the total profit (Sales income – All the production costs) by taking the precise decision sending the pigs to the slaughterhouse when it is their perfect time and afterwards, selling them. Meanwhile it must be taken into account the pigs’ weight, the pigs’ intake, the different the maintenance costs of the pigs, the transportation cost and also, the payment matrix of the slaughterhouse.

2.1.3. MODEL FORMULATION

The indexes, parameters, decision variables used in this model, as well as the objective function and its constrains will be presented in this section. Apart from that, it will also be explained the options the farmer can do when running the model: optimization or sensibility analysis.

Indexes

Most of the operations on the farm are planned on a weekly basis, so the fattening stage is divided into weekly periods, T.

The indexes of this fattening model are presented below:

t = 1, 2,. . .[T], index of weeks into which the fattening period is divided.

i = 1, 2,. . .[P], index of partitions to cluster pigs into growth categories.

k = 1,2. . .[K], index of kinds of truck.

Parameters

The parameters of this fattening model are presented below:

N = batch size, representing the number of pigs introduced to the fattening farm.

$\boldsymbol{n_i}$ = cluster of Growth category *i*, in which the initial batch was partitioned.

$\boldsymbol{\bar{w}_{it}}$ = mean value of the liveweight of pigs (kg) in the Growth category *i* at Week *t*. We assume that the liveweight of the batch follows a normal distribution, $w_t \sim N\ (\mu_t, \sigma_t)$. (see *Table Annex 1*)

ω = selling price, €/kg of carcass weight[1].

$\boldsymbol{\bar{f}_{it}}$ = cumulative feed intake average (kg) by a pig in Growth category *i* until Week *t*. (see *Table Annex 1*)

$\boldsymbol{\beta_{it}}$ = bonus given by the slaughterhouse (€/kg of carcass weight) as a function of Growth category *i* at Week *t*. (see *Table Annex 2*)

$\boldsymbol{\delta}$ = cost (€) per kilogram of feed intake.

[1] Carcass weight: the weight of the slaughtered pig's cold body, either whole or divided in half along the mid-line, after being bled and eviscerated and after removal of the tongue, bristles, hooves, genitalia, flare fat, kidneys and diaphragm. [2]

$\boldsymbol{\lambda_k}$ = fixed cost in euros for truck of Type k sent to the slaughterhouse.

$\boldsymbol{\alpha_i}$ = cost in euros for other expenses such as vet and medicines which is constant throughout the weeks and only depends on the category i.

$\boldsymbol{\xi}$ = cost in euros per young pig purchased.

$\boldsymbol{\psi_k}$ = capacity of trucks of Type k in number of pigs.

$\boldsymbol{\tau_k}$ = capacity of trucks of type k in kilograms of load.

Decision variables

The decision variables of this fattening model are presented below:

$\boldsymbol{x_{it}}$ = number of pigs from Partition i to be sent to the slaughterhouse in Fattening week t.

$\boldsymbol{z_{it}}$ = inventory of pigs from Partition i at the beginning of Fattening week t.

$\boldsymbol{y_{kt}}$ = number of trucks Type k needed at Week t for shipping pigs.

$\boldsymbol{h_{it}}$ = binary variable taking value 1 when pigs from two consecutive partitions (i – 1, i) are sent to the slaughterhouse, 0 otherwise.

$\mathbf{d_{it}}$ = binary variable taking value 1 when pigs from pigs from Partition i at Week t are sent to the slaughterhouse, 0 otherwise.

Objective function

The objective function of this fattening model are presented below:

$$Max\ g = \sum_{i=1}^{|P|}\sum_{t=1}^{|T|}(\omega + \beta_{it}) * \bar{w}_{it} * x_{it} - \sum_{i=1}^{|P|}\sum_{t=1}^{|T|}\delta * f_{it} * x_{it}$$
$$- \sum_{t=1}^{|T|}\sum_{k=1}^{|K|}\lambda_k * y_{kt} - \sum_{i=1}^{|P|}(\xi + \alpha_i) * n_i$$

Model constrains

This fattening model includes some constrains which are written below:

$\boldsymbol{n_i = z_i} \, \forall \boldsymbol{i} \in \boldsymbol{P},$ fixes the initial inventory of pigs per Growth category i at Week 1.

$\boldsymbol{x_{it} \leq z_{it}} \, \forall \boldsymbol{i} \in \boldsymbol{P}, \boldsymbol{t} \in \boldsymbol{T}$, refers to how deliveries are limited by current inventory.

$\boldsymbol{x_{it} \geq z_{it} - N(1 - h_{it})} \, \forall \boldsymbol{i} \in \boldsymbol{P}, \boldsymbol{t} \in \boldsymbol{T}$, determines the binary variable $h_{it} \in \{0,1\}$. It establishes that all the pigs in the *i*-growth category at Week *t* have to be sold before those from the lighter growth category (i – 1) can be selected; that is, pigs of higher weight are sold first.

$\boldsymbol{z_{i|T|} - x_{i|T|} \leq 0} \, \forall \boldsymbol{i} \in \boldsymbol{P},$ updates the inventory for each category for the following week.

$\boldsymbol{z_{it+1} = (z_{it} - x_{it})} \, \forall \boldsymbol{i} \in \boldsymbol{P},$ states that all the pigs in the batch must be sold at the end of the Week |T| according to the AIAO management strategy.

$\sum \boldsymbol{x_{it}} \leq \sum \boldsymbol{\psi_k} * \boldsymbol{y_{kt}} \, \forall \boldsymbol{t} \in \boldsymbol{T} \qquad \sum \boldsymbol{\overline{w}_{it}} * \boldsymbol{x_{it}} \leq \sum \boldsymbol{\tau_k} * \boldsymbol{y_{kt}} \, \forall \boldsymbol{t} \in \boldsymbol{T}$

These two constrains in these two equations determine the number of trucks of each type needed to deliver pigs to the slaughterhouse, taking into account the capacity of trucks in terms of number of animals and kilos of load.

$\boldsymbol{x_{it} \leq z_{it}} * \boldsymbol{d_{it}} \, \forall \boldsymbol{i} \in \boldsymbol{P}, \; \boldsymbol{t} \in \boldsymbol{T}$, introduces an auxiliary binary variable, dit 2 {0,1}, intended to detect whether a group of pigs in Growth category i at Time t is sent to the slaughterhouse, or not.

$\boldsymbol{h_{it} \leq d_{it}} \, \forall \boldsymbol{i} \in \boldsymbol{P}, \boldsymbol{t} \in \boldsymbol{T} \qquad \boldsymbol{d_{it} + d_{i+1t} \leq 1 - h_{i+1t}} \, \forall \boldsymbol{i} \in \boldsymbol{P}\{|\boldsymbol{P}|\}, \; \boldsymbol{t} \in \boldsymbol{T}$

These constrains in these two equations link the binary variables h_{it} and d_{it} and force the delivery of heavier pigs first and prevent the delivery of pigs from nonadjacent categories.

2.1.4. LINEAR PROGRAMMING AS A DECISION TOOL

Basic Situation – Optimization

This model has been designed in order to know the best time to send the pigs to the slaughterhouse so that pig-production companies are able to maximize the total profit the company related to pigs production, while supporting marketing decisions on farms without using the common rule-of-thumb which is being made at the moment in this sector.

This model based on vertically integrated companies, which operate with an AIAO management policy. Throughout the fattening of the animals, the farmer is the person who visually chooses which animals should go to the slaughterhouse, as it is just mentioned above with the rule-of-thumb. Once the animals are selected, they are sent to the slaughterhouse with trucks provided by the same slaughterhouse. Transportation costs are organised by the slaughterhouse and deducted from the payment afterwards made to the farmer. Therefore, the farmer can estimate what would be more or less the economic result obtained from that shipment, since the farmer will have been informed previously of the pricing-grid reward system and the sale price per animal. Each slaughterhouse has a price, but there is a price reference that is what Mercolleida (Spain) auction market sets. Thus, slaughterhouses estimate around that value.

Regarding the pricing-grid reward system follows the SEUROP classification table, approved by the Commission Regulation (EC) No 1249/2008 of 10 December 2008 in which is written detailed rules on the implementation of the Community scales for the classification of beef, pig and sheep carcases and the reporting of prices thereof [3].

% lean	Classification	Bonus (€/kg of carcass)
>60	S	+0.12
55 to 60	E	+0.07
50 to 55	U	0.00
45 to 50	R	–0.07
40 to 45	O	–0.12
<40	P	–0.30

Table 1: SEUROP classification table

In *Table 1*, we observed that what is interesting is to have as much carcass weight as possible as well as so much lean weight[2]. In other words, the best situation for a farmer is to find animals that generate so much meat without fat or just a little (classified with an S).

The parameters and variables to run this model will be presented below:
The total number of pigs in a lot (N) is 1000 with an initial average weight of 29.7 kg and a deviation of 3.9 kg. Considering that 17 will be the number of weeks the batch will be in the fattening farm.
Once entered into the fattening farm, the growth stages (P) is divided into 10 groups or partitions. Therefore, each cluster of growth category is 100 pigs. For the present study, the biological parameters related to pig growth and feed consumption corresponded to a crossbreed of *(Large White x Landrace) x Piétrain*. In *Table Annex 1*, it can be seen the growth of the pigs as well as the amount of food eaten per week.
During the their stay at the fattening farm, costs are applied to these pigs such as the purchase price per standard average piglet (around 20kg) is 40.55€, the cost of the food which is 0.28 €/kg and other costs such as medicines, vet and water around 21€ per pig. Regarding transportation cost is defined 475€ per truck with a maximum capacity of 240 pigs each truck. Thus, this is the cost of every shipment.

Sensibility Analysis

In the same way that the farmer can execute the optimization, the farmer is also able to execute a sensitivity analysis. As mentioned above, decision-making in the pig sector is becoming increasingly complex. Therefore, the advantages that the farmer can have to improve their decisions by executing the model with a sensitivity analysis are very wide.
Through this sensitivity analysis, the farmer can see what impact certain variables have on the company's final benefit. These variables just mentioned would be all the costs related to the fattening process, which are the purchase cost per piglet, the cost of the food, medicines or the shipping cost per truck to the slaughterhouse. Besides this, a

---

[2] Lean weight: basically the meat that there is, meaning the complementary part to the fat.

variable which can also have a impact on the optimal marketing policy is the capacity of the truck.

Farmers, based on this sensibility analysis by choosing the variables they desire, can make better decisions in order to maximize the total benefit of their company.
In this project, some of these variables will be analysed to see the impact they have on this final benefit.

2.1.5. MODEL SOLVING AUTOMATION

In this section the automation of the model execution will be explained. This way, the main objective is to reduce complexity so that anyone can easily use the model and the results that can be obtained through it. Since the objective is to integrate the model with a decision support system, this automation was necessary and was a prerequisite.

In *Figure 4*, it can be seen how the automation of the model has been implemented with Python. As it has been commented in *section 2.1.1*, Pyomo has been used as a modelling language and as a consequence, complexity has been reduced.

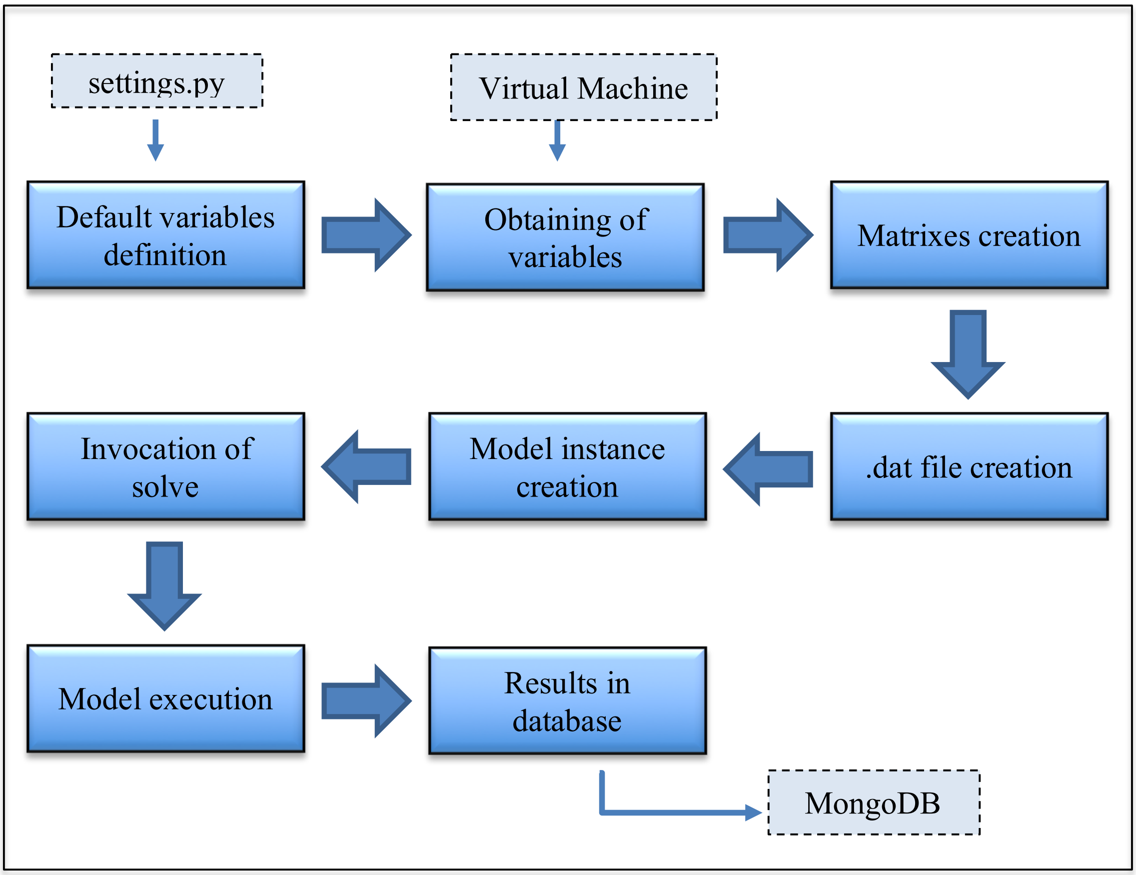


Figure 5: Diagram of implementation in Python
Source 5: Own

In this implementation the parameters of the model that the user can not modify are defined in the "settings.py". On the another hand, the input parameters of the script are the ones that each user can modify depending on their characteristics and thanks to these parameters, the growth and consumption tables of the animals can be adjusted.

These tables are important since they would vary depending on the conversion rates, average daily gain and entry weight of the batch. This last variable is the key to measure the length of these tables. The model works by default with 17 weeks. However, depending on the entry weight of the batch, which is provided by the user, the table will be recalculated after measuring the weeks of the batch.
A good adjustment of the tables implies obtaining a result of the optimal model.

The automation of the model solving involves a thorough and accurate processing of the information. Firstly, the default variables are defined. These variables are imported from a configuration file, which is called “settings.py”, so that it can facilitate the handling of the data and a possible modification of it in the future. Moreover, these settings are based on a previous calibration of the farm and are crucial to define a better image of the current situation of the farm. Currently, is technical information that farmer do not know.
Secondly, the necessary parameters are obtained for the model to work through a virtual machine with the relevant command. Afterwards, through functions taking into account all the necessary parameters, relevant matrices of this process are generated, which are: the weight matrix, the growth matrix and the slaughterhouse bonuses. After the generation of these matrices and once we have all the data update in regards to the different variables, the .dat file is created. In this file, all the data that the model will use will be written. Once we have the data stored in the file, an instance of the model is created to treat with the model and is the one that will be used throughout the program. By creating this instance, it is produced a merge between the mathematical formulation with the data recently stored. Next step is to select the solver (in our case: CPLEX) and solve the model by the invocation of solve.
Finally, the results obtained from the execution of the model are stored in the database. MongoDB is the database chosen for this project, as it will be explained in the next section.

2.2. BACKEND AND DATABASE

2.2.1. TECHNOLOGIES

Throughout the design and implementation of the backend and the database, the tools, which have been used, have been: Webstorm, NodeJS, MongoDB and Postman.

Figure 6: Logo of Webstorm, MongoDB and Postman

Source 6: Official websites

Webstorm, javascrip IDE developed by JetBrains, is the one that has been chosen for working on the backend development. Webstorm provides code analysis in order to provide the best code completion. Webstorm takes the code completion for Angular JS apps. No matter if you work on a client-side or on the server side (Node) applications, Webstorm also provides an advanced debugger, a unit tester and an efficient tracer.

The backend developed in this project is monolithic and has been coded with NodeJS and expresses all the models and internal routes of the application. Another important function which the backend has, in regards to the executions of the model, is to obtain all the data in order to create an execution. Once it is created, it processes with the connection with a micro service that will manage them.
At backend, it will be generated a zip file that will contain a "run.sh", the command with the input parameters of the script which are needed to run the program in Python and after this command, it will also contain a very important element; *poweroff*. This last element works to destroy the virtual machine, which is created by a micro service on the cloud platform, once the model has been executed.

The management of the database is done through CRUD operations. The acronym CRUD stands for Create, Read, Update and Delete. These are the four basic operations of persistent storage. These operations have been adapted for either the management of the data or the use of the application.

In addition to this, the backend also equips a tight integration with VCS (Version Control System). Though this project, the chosen VCS is Github. What is also interesting is that this backend is protected, using a security of a token type. A token is used for each request, which means that it is not open to all audiences, but to registered users.

MongoDB is a data base which is placed in the cloud. MongoDB is a non-relational database system oriented and deployed a dynamic schema in which has being used JSON-like documents in order to make the integration of data clear and faster. A non-relational database has been chosen due to its advantages over relational databases. On one hand, non-relational databases can offer flexibility (lack of rigid structure), good fit for modern JavaScript frameworks (direct use of JSON), big data processing and real time statistics/data analysis. On the other hand, relational databases provide strict data integrity enforcement, reliable way of combining the records during fetching and less scalability than non-relational databases.

In order to facilitate the interaction between the data models and the MongoDB, the *mongoose* library has been used.

All data schemas are structured following the same way: key-value. For instance, the scheme of the fattening farm has the following content.


```
_id: ObjectId("5cac9edf52df03068438e624")
name: "eFatMireia"
capacity: 1000
stock: 100
type: "fattening"
latitude: 434534535345
longitude: 434534535345
piglets_price: 50
other_costs: 21
intake_cost: 0.3
owner: "5b335cec5bea543654c5f2eb"
updatedAt: 2019-05-18T18:44:28.170+00:00
```


Figure 7: Fattening farm data schema
Source 7: Own

Postman, API development environment which integrates several tools for managing APIs. Through Postman, the RESTful API can be tested by creating request associated with GET, POST, PUT, DELETE.

REST requests

The API of this project responds to the next subset of routes, which are the most important in this project.

Operation	Route	Description	Token Required
POST	/fatfarms/create	Creation of a fattening farm data schema with their corresponding information.	Yes
GET	/fatfarms/show/:id	Visualization of the farm data schema with the identifying *id* of the farm.	Yes
GET	/fatfarms/show/:userId	Visualization of the farm data schema with the identifying *userId* of the farm.	Yes
PUT	/fatfarms/update/:id	Modification of the farm data and the route has the farm *id*. Data that can be edited by the user through the client interface.	Yes

Table 2: Rest request operations

2.3. FRONT-END

2.3.1. TECHNOLOGIES INVOLVED

Angular 2 is used for developers to build applications on the web. It is a TypeScript based on open-source front-end web framework that is powered by Google and by a community of individuals and corporations. Angular2 works with HTML and CSS. Moreover, it has another useful utility that is Angular CLI Server with which developers can start building fast, add components and test. Therefore, Angular CLI is a powerful tool whose aim is though HTTP requests, interact with the server. All projects run on this default server: *http://localhost:4200*. This address is used to test the front-end in local. However, at the end, it will be deployed to a production server that will be like a common IP, more precisely on *http://stormy02.udl.cat*.

Figure 8: Logo Webstorm and Angular
Source 8: Official websites

2.3.2. FUNCTIONALITIES

The client interface has been developed in Angular, as I have mentioned in the previous section. The main aim is make this interface become a perfect environment where the user can interact with the model and also, with their data regarding their farm. Therefore, following these aspects, the interface has to be able to store the data and try to make the user enter the minimum information to run the model.

Throughout this section, the main functionalities of this client interface will be explained. However, not all have been able to become functional. For this reason, this section will be divided into two parts: the current functionalities of the interface and the future functionalities that the interface may have which are already designed.

Current functionalities

User Login

It works like other user login operation. The user types the username and password. Doing this, the user is making a petition to the API with the credentials entered. Once it they are accepted implies that the entered data is correct, therefore, the user can have access to the interface. After a successful authentication, the server generates a unique token for the user, which is required to perform any of the operations that require to be logged, for instance, running the model or visualising the farms that the user has. This token expires either after 24 hours or when the user logs out.

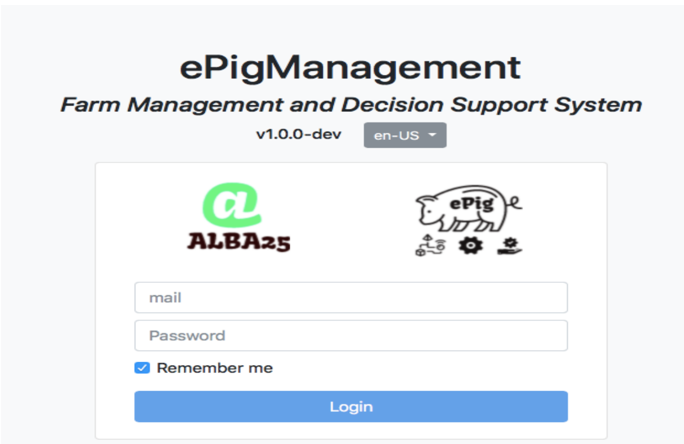


Figure 9: Interface - User Login
Source 9: Own

View and Edit farm information

Users can see information about the farm and also the costs associated. Besides this, users have the functionality to edit the provided information whenever they want and afterwards, this new data is stored into the data base.

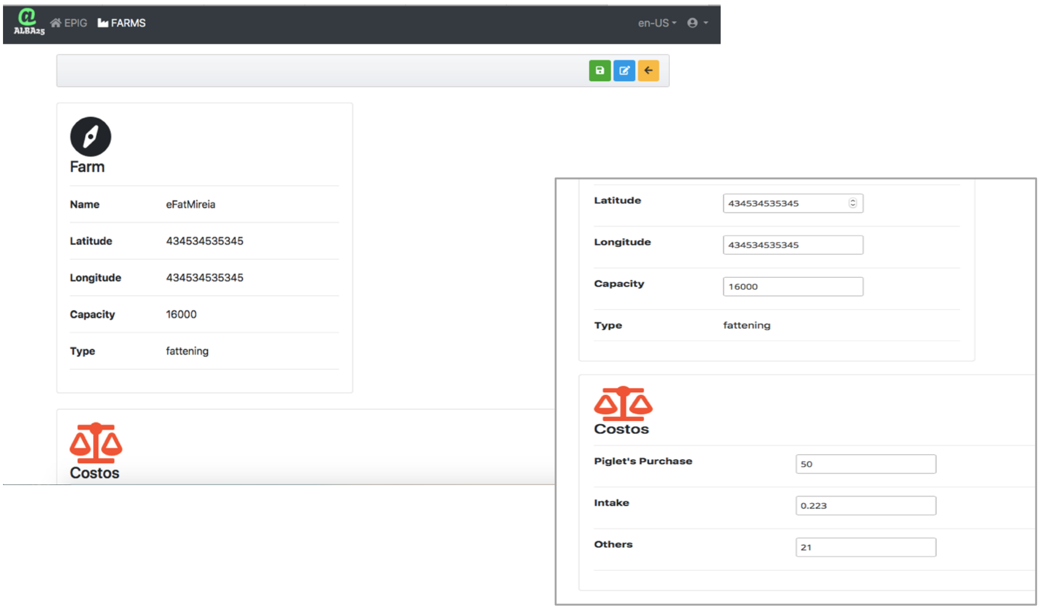


Figure 10: Farm information with edit mode
Source 10: Own

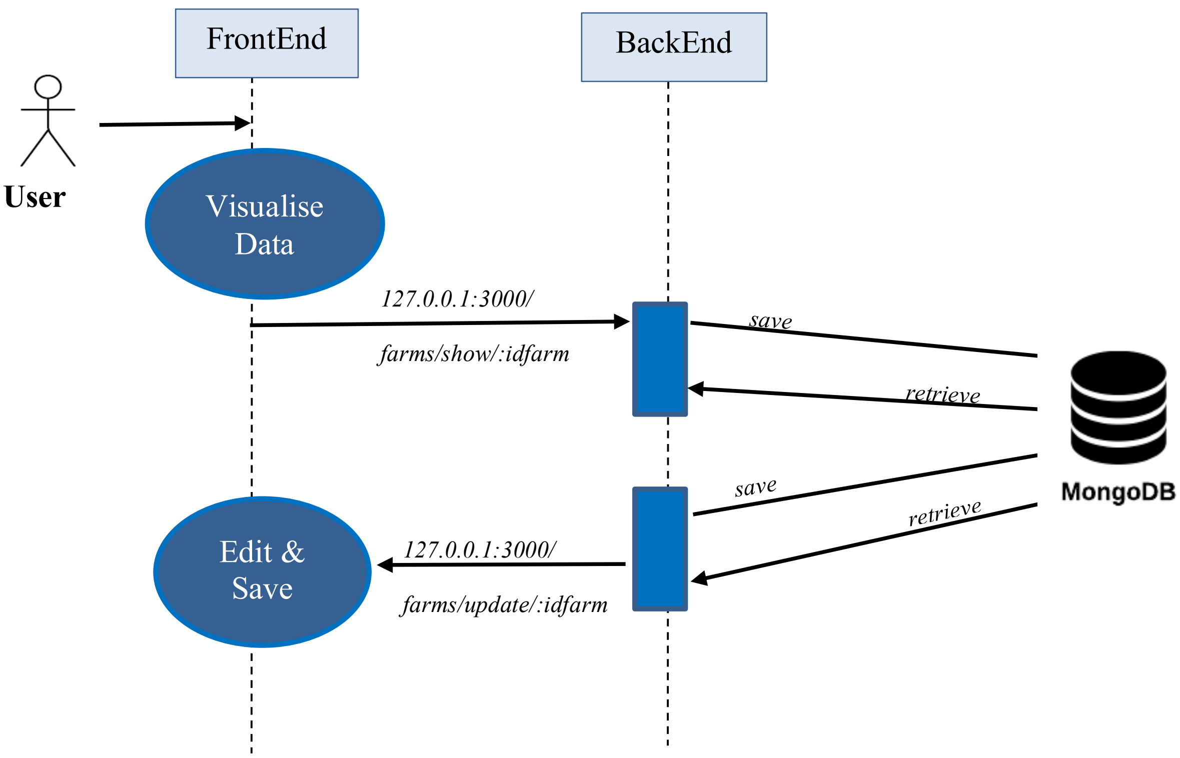


Figure 11: Farm information sequence diagram
Source 11: Own

Execution Model

The user will be able to run the model either choosing an optimization operation or a sensibility analysis. So far, only these two operations can be executed on the model at the client interface. On one hand, running the optimization model means showing to the user his profit, cost and benefits. On the other hand, running a sensibility analysis, the user is allowed to choose which is the parameter that he wants to execute and test the sensibility of that. Therefore, it will be provided the variability of profit, cost and benefit, those that the user will have by changing the value of the parameter chosen.

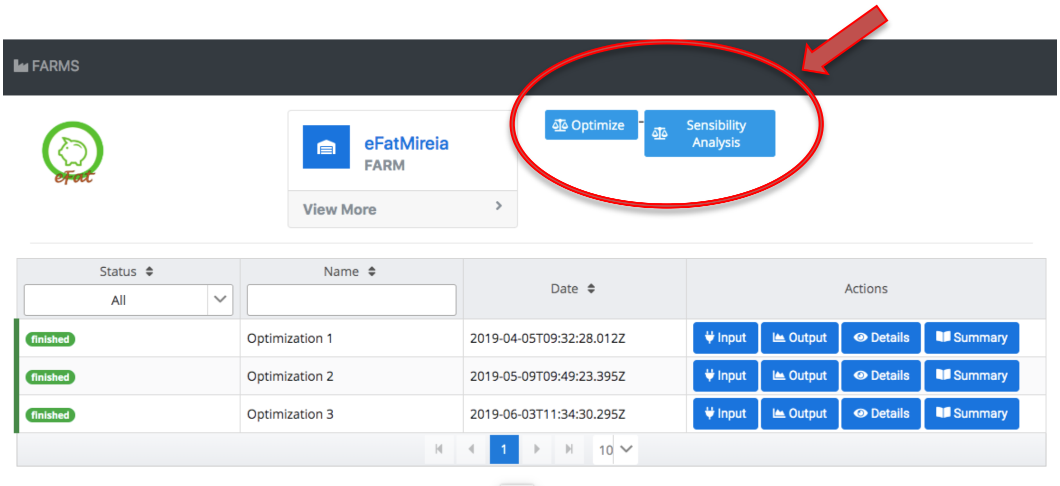


Figure 12: Optimization operation or sensibility analysis button display
Source 12: Own

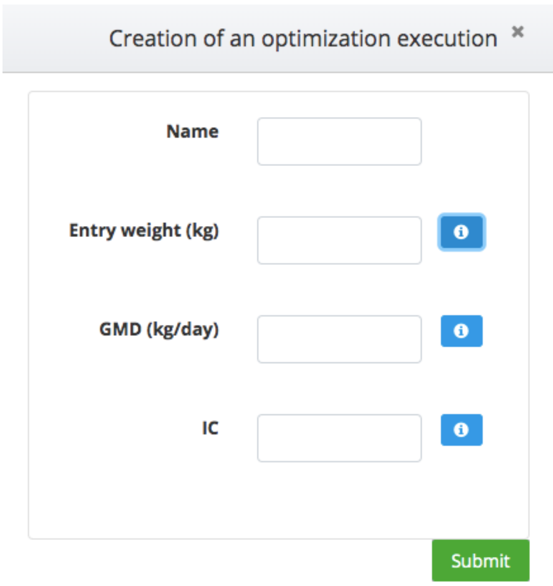


Figure 13: Creation of an optimization execution
Source 13: Own

Visualization of the results after the execution of the model.

The interface provides a section where can be seen the different results of the model. On the hand, the economic results, which that farm would get, are provided. On the other hand, it can be seen how many animals have been sent to the slaughterhouse and how many trucks have been used per week. Moreover, a graph is provided where can be seen the mentioned data in a visual way.

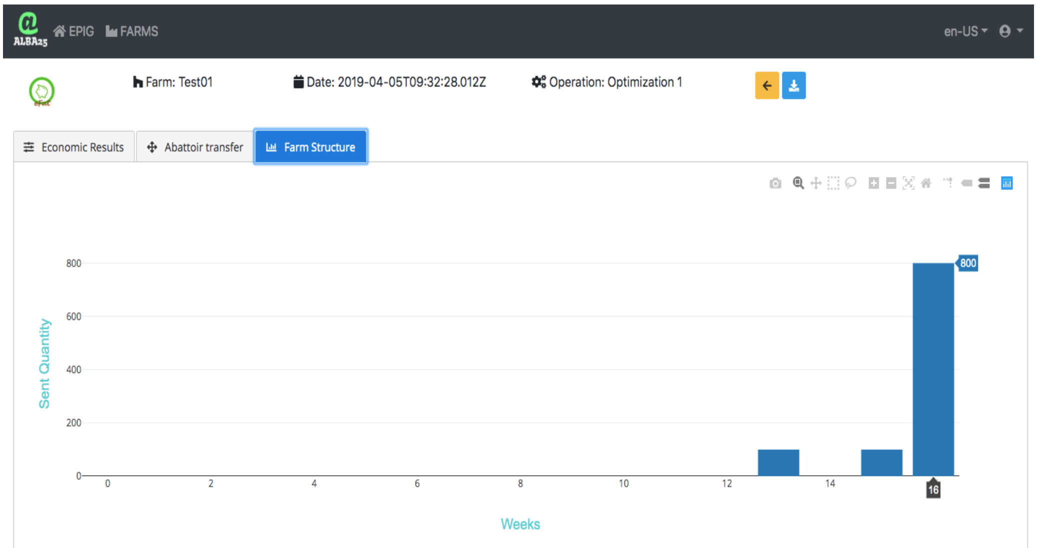


Figure 14: Delivery of fattened pigs to the slaughterhouse display

Source 14: Own

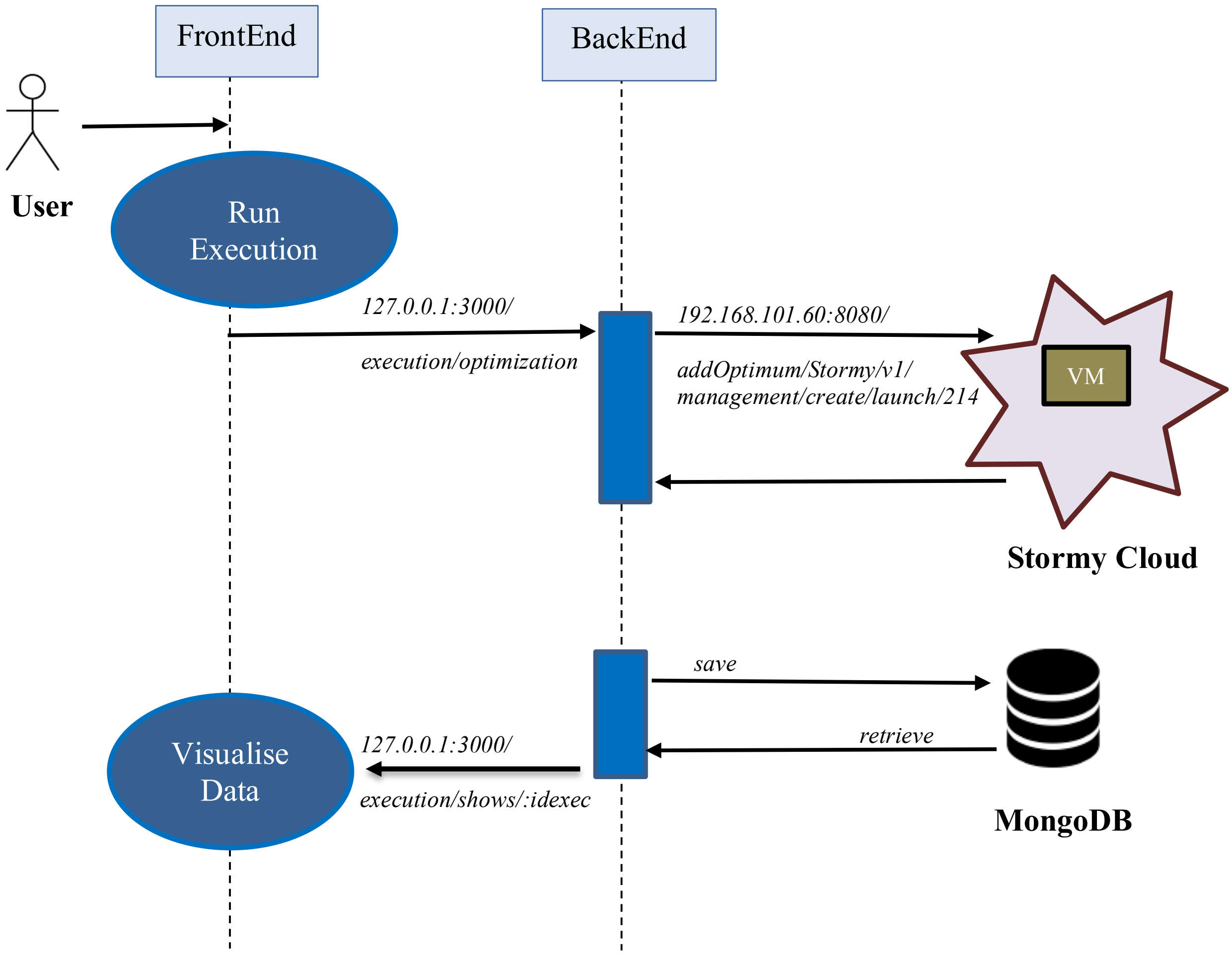


Figure 15: Execution results visualization sequence diagram

Source 15: Own

View operations history

On the main page the user is provided with the history of their operations in which they can see the results obtained, the parameters entered in said operation. As well, the state of that operation, being a completed operation or operation that gave an error or that is still in progress.

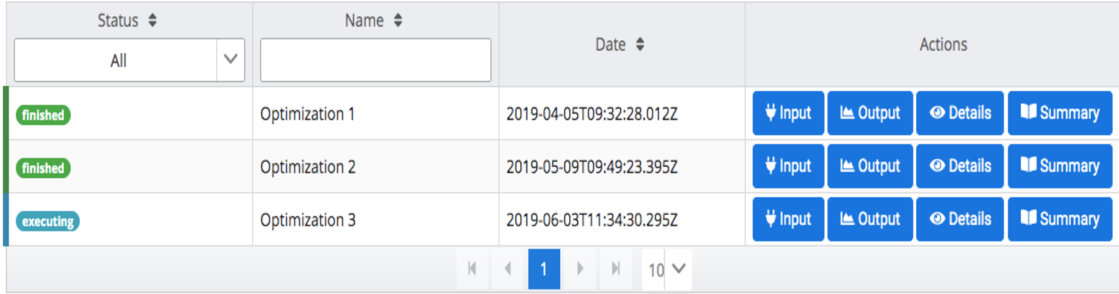


Figure 16: Operations history display

Source 16: Own

Future designed functionalities

View trucks and slaughterhouse information

The interface provides a table to the user where can be seen how many trucks the farm has contracted or the trucks that the farm uses. Each truck with its particular information. In addition, it also provides another table with information of the slaughterhouses associated with that farm. As in the trucks table, the information of each slaughterhouse is also available, so that the farmer can compare and decide which is the best option.

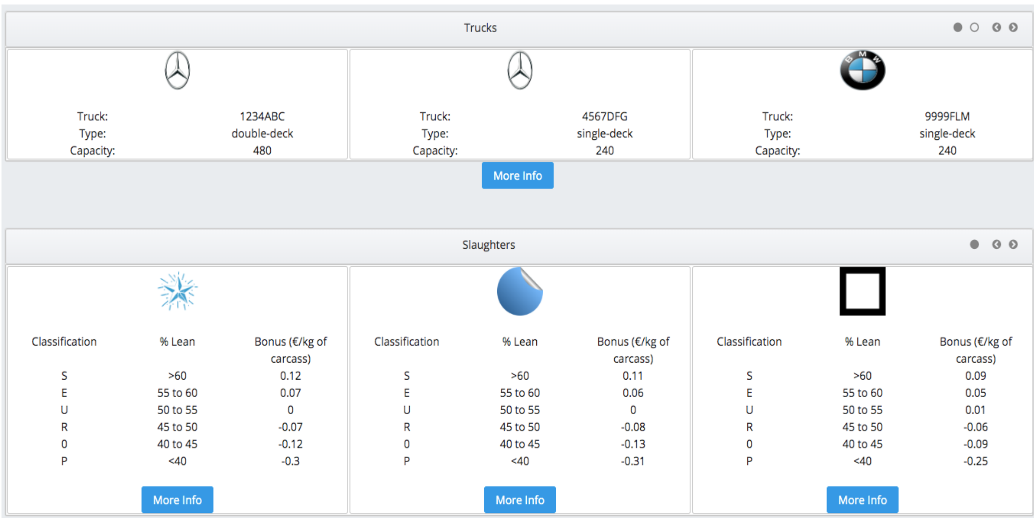
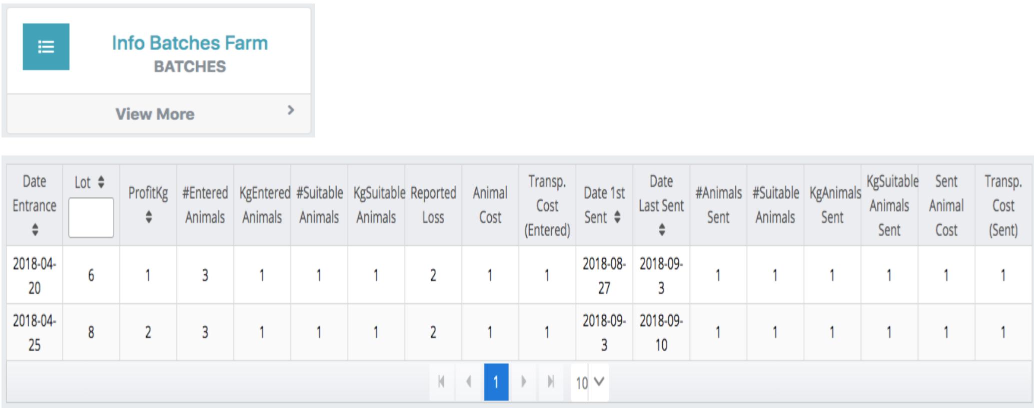


Figure 17: Trucks and slaughterhouses information front-end

Source 17: Own

View batches information

The interface enables the user to see the information in regard to all the batches that have arrived at the farm. For instance, entrance and sent date, total entered and sent kilograms, several costs and other information more that can be seen on the *Figure 9*.



Date Entrance	Lot	ProfitKg	#Entered Animals	KgEntered Animals	#Suitable Animals	KgSuitable Animals	Reported Loss	Animal Cost	Transp. Cost (Entered)	Date 1st Sent	Date Last Sent	#Animals Sent	#Suitable Animals	KgAnimals Sent	KgSuitable Animals Sent	Sent Animal Cost	Transp. Cost (Sent)
2018-04-20	6	1	3	1	1	1	2	1	1	2018-08-27	2018-09-3	1	1	1	1	1	1
2018-04-25	8	2	3	1	1	1	2	1	1	2018-09-3	2018-09-10	1	1	1	1	1	1

Figure 18: Batches information front-end

Source 18: Own

2.4. ARCHITECTURE

The architecture of this project is presented in this section and how each of the parts of it connect to each other. This architecture is composed of three layers: presentation layer, data source and domain logic. The presentation layer comprises the logic to handle the interaction between the user and the application. The data source has to do with communication with other data systems that carry out tasks on behalf of the application. The domain logic is the specific functionality that the application must do for the work domain.

One of the advantages of this architecture is the use of a private cloud that provides elasticity. This private cloud platform used to implement this architecture is Stormy[3].

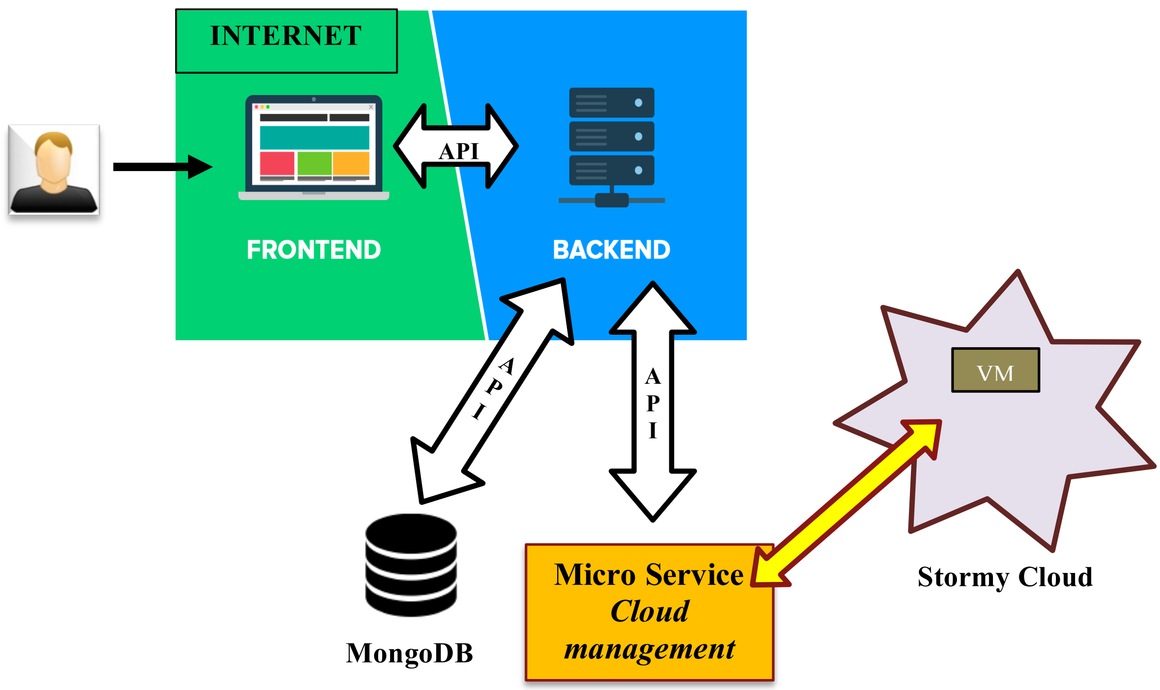


Figure 19: Project architecture
Source 19: Own

After having described the three layers, it will be explained in more detail the process involved in this architecture.

[3] Stormy support site is *https://stormy.udl.cat*

There is the user who connects via Internet to the browser (frontend). The other important elements of this architecture are a backend, a private cloud (Stormy), a database (MongoDB) and a micro service that manages virtual machines (creation, deployment, task execution and destruction) at Stormy platform using the OpenNebula backend.
The frontend communicates with the backend through an API. Additionally, the second one is also communicated through an API with the micro service and with the database in order to store or retrieve data. This micro service what it does simultaneously is: create resources, execute the model or the automation and destroy the resources which have been already created. Destroying the virtual machine is due to avoid the problem of not having enough computational resources on the cloud platform. Another aspect to take into account is that the model is on a NFS (**N**etwork **F**ile **S**ystem) server. In order not to create a .zip file every time (mentioned in *section 2.2.1*), all the virtual machines that are created on the cloud platform are connected to an NFS. NFS is a distributed system that allows to access and save remote files to a network environment. Therefore, it is a persistent system, where any modification or query can be carried out on the files that it contains.

This micro service is already developed and it has been integrated into the functionalities of this project. Regarding the backend and the frontend are explained in sections *2.2* and *2.3* respectively.

Another main feature of this application is that it creates and destroys resources on demand. Meaning, only resources will be active when they will be need and used, otherwise they will be destroyed to alleviate the computational resources. Integrating the mentioned micro service and mounting the executions, what has been achieved is not loading these executions on the backend. The backend can not withstand such a huge amount. The reason is that the model also had to be run and the machine on the backend should have been so powerful. An hypothetical case of an arrival of a large number of requests and a go-down of this machine, it would also mean that the application would go down. However, this architecture is designed on a way that the requests go to the micro service and this one handles them.

2.5. PROJECT MANAGEMENT

2.5.1. SCRUM METHODOLOGY

For this project, it has been followed the idea of the Scrum methodology which is an agile technique to maximise the product and project value.

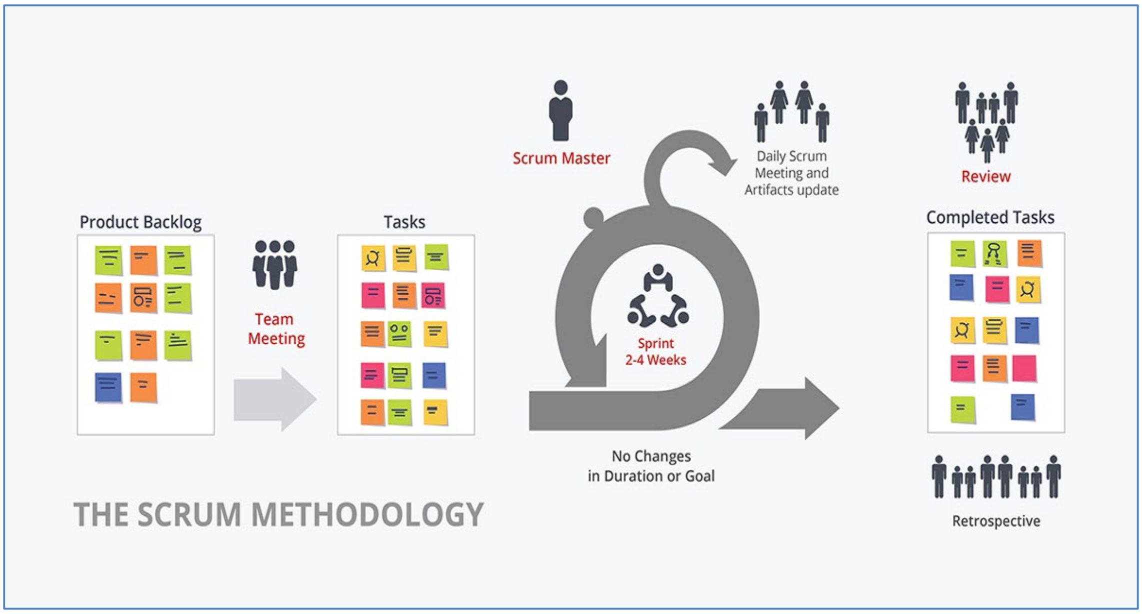


Figure 20: Overview of the scrum methodology
Source 20: Systems Valley's website

All the steps of a Scrum framework follow are the following ones:
Firstly, the product owner creates a prioritized product backlog. The product owner will put the most important items on the top and the least important at the bottom. This may change in every spring or the prioritized order Secondly, the spring planning is done in which the amount of work per spring is planned. Thirdly, Scrum Meetings are done. The team uses this time to determine how best to share information and to help each other in order to complete their spring commitment. This short meeting exposes risk and knowledge to be shared which in turn help the team to be more effective. Up to this point, one important person is the Scrum master. The person filling this role is responsible for helping ensure the success of the project. They remove impediments for the team, help the team make responsible decisions and basically support the team in any way possible.
At the end of the sprint, two other meeting are held, the first is the spring review and the second is the Spring Retrospective. The first one is basically to improve the product

with the feedback of the client by showing what the team has accomplished and the second one is to improve the team.

By seeing the Scrum framework, we can have an idea how this methodology works. We have done the same by making a sprint every week. Each sprint shows the work accomplish and those can't be completed in the actual sprint. For this project, we have to different roles:

Product Owners: Adela Pagès, Lluís M. Pla and Esteve Nadal.

Scrum master: Jordi Mateo.

There are many advantages of implementing SCRUM such as the decrement of the development time and also the agility and flexibility which this methodology offers. Adaptability to the newer requirements and demanded modifications.

2.5.2. WORKFLOW

The project has been divided in different sprints, once one sprint was successfully completed, the following one was carried out. Mainly, these sprints are related to the process of developing the client application and the API that is where the optimization model of optimization plays its role.

Sprint 1: Familiarization of the environment to work with the model.

1. Be familiar with the model and its Excel macros and read some articles about it.
2. Be familiar with the cloud platform.
3. Be familiar with CPLEX and linear programming.
4. Understand the model and its constraints.

Sprint 2: Realization and evaluation of formulas.

1. Propose a formula that only knowing the average daily gain can recalculate the table.
2. Propose a formula that can only be recalculated by only knowing a conversion rate.

Sprint 3: Evaluation of the data model.

1. Propose a data model to store all the information in a DB.

Sprint 4: Front-end preparation.

1. Be familiar with Angular 2 and prepare a prototype of a front-end.
2. Make the pre-design front-end for a fattening farm.

Sprint 5: Python automation.

3. Script with python to automate the calculations to get the tables with the proposed formulas.
4. Script with Python to automate the model.

Sprint 6: Python analysis of sensibility.

1. Script with Python to automate the analysis of sensitivity of a parameter.

Sprint 7: Connection of the model automation with the database.

1. Script with python to automate the resolution of the model using the data stored in MongoDB.

Sprint 8: Creation of the back-end and communication with the front-end.

1. Develop and prepare the back and front-end in order to have a future connection.
2. REST server (Nodejs) to perform CRUD (Create, Read, Update, Destroy) of all defined data models.

Sprint 9: Creation of an execution in the back-end and establishment of full connection with the cloud.

1. Develop the form to obtain the parameters of the execution provided by the farmer and store it in the DB.
2. Establish the connection with the cloud and launch the execution in the cloud.

Sprint 10: Display optimization model results.

1. Display the results through tables and graphic.

Sprint 11: Analysis of the results obtained after the execution of the model.

1. Results with (real or simulated) data.

Sprint 12: Writing the report.

3. RESULTS AND ANALYSIS

Throughout this section, it is shown an analysis of results depending on the modified variable or parameter used by the model. This analysis is focused on the economic side that is the part that has more interest to the farmer. Therefore, the aim is to show in a clear and understandable way, the results that can be extracted from the model according to the parameter that is modified.

3.1. MODEL PARAMETERS AND RESULTS

The model assumes a batch of pigs of the same age, a mean weight and a standard variation. A regression growth curve from experimental data is used for the estimation of the weight distribution of the batch.

On an ascending scale, if the coefficient has a value of 1, it means that the batch has the the regular variation the model considers. As closer to 0 is the coefficient, more homogeneous are the animals of the batch. On the contrary, the larger this coefficient is, more heterogeneous the batch is and, thus, the larger the weight and the intake variability of the animals in this batch are.
The model uses coefficient 1 as a default variable, but it can be modified depending on the farmer's conditions. Moreover, the default tables of weight and intake, used by the model, can be found in *Table Annex 1*.

Finally, the default parameters used by the model will be presented in the following *Table 3*, unless the farmer has others and wants to modify the following ones.

PARAMETER	***VALUE***	***PARAMETER***	***VALUE***
Weeks	17	***Other costs (€)***	21
Average Daily Gain (kg/day)	0.6983	***Truck cost (€)***	475
Conversion Rate	1.223	***Truck capacity (nº pigs)***	240
Partitions (nº partitions)	10	***Farm Capacity (nº pigs)***	1,000
Food Cost (€)	0.28	***Animals Stock (nº pigs)***	100
Piglet's purchase cost (€)	40.55	***Animals / each group***	100

Table 3: Model default parameters

From these mentioned parameters, the following optimization result is obtained after execution:

'Profit': 147,242.67
'Cost': 51,809.52
'Total': 95,433.15
'Weeks': [{
'trucks': 0.0
'animals': 0.0
}, {
'trucks': 0.0
'animals': 0.0
}, {
'trucks': 0.0
'animals': 0.0
}, {
'trucks': 0.0
'animals': 0.0
}, {
'trucks': 0.0
'animals': 0.0
}, {
'trucks': 0.0
'animals': 0.0
}, {
'trucks': 0.0
'animals': 0.0
}, {
'trucks': 0.0
'animals': 0.0
}, {
'trucks': 0.0
'animals': 0.0
}, {
'trucks': 0.0
'animals': 0.0
}, {
'trucks': 0.0
'animals': 0.0
}, {
'trucks': 0.0
'animals': 0.0
}, {
'trucks': 0.0
'animals': 0.0
}, {
'trucks': 1.0
'animals': 100.0
}, {
'trucks': 0.0
'animals': 0.0
}, {
'trucks': 1.0
'animals': 100.0
}, {
'trucks': 4.0
'animals': 800.0
}]

These results show the benefit and costs that the farmer would have with a batch of these conditions mentioned above and with these same costs. Therefore, this model enables the farmer to know if the company has benefits or losses. In addition, it shows which is the week and the amount of pigs that have to be sent to the slaughterhouse. The number of trucks to be used are also shown after executing the model.

In a short summary, if everything follows the default numbers of this model, we would obtain:

PROFIT	*COST*	*TOTAL*
147,242.67 €	51,809.52 €	95,433.15 €

Table 4: Economical results from default parameters

3.2. ANALYSIS COMPUTATIONAL RESULTS

The homogeneity of a batch is an important key factor for the farmer. In this section, conclusions will be drawn according to the results obtained after the execution of the model.
Homogeneity is known as equality or similarity in the nature or gender of several elements. Therefore, for a farmer, the most favourable situation is obtaining a batch without variability, since the batch will be more controlled.

The analysis is based on changing the percentage of the homogeneity of the batch and see how much impact this has on the results of the model. The variables and parameters used for this analysis can be found in *Table 3* of *section 3.1*.

After running the model with different homogeneities (coefficients), these are the economical results obtained:

COEF.	*PROFIT (€)*	*COST (€)*	*TOTAL (€)*
0	156,740.24	51,334.52	105,405.72
0,25	154,542.90	51,334.52	103,208.38
0,5	153,540.46	51,334.52	102,205.94
0,75	148,537.33	51,334.52	97,202.81
1	147,242.67	51,809.52	95,433.15
1,25	143,458.56	51,334.52	92,124.04
1,5	139,160.13	51,809.52	87,350.61
3	123,374.75	52,284.52	71,090.23

Table 5: Economical results by a sensibility analysis

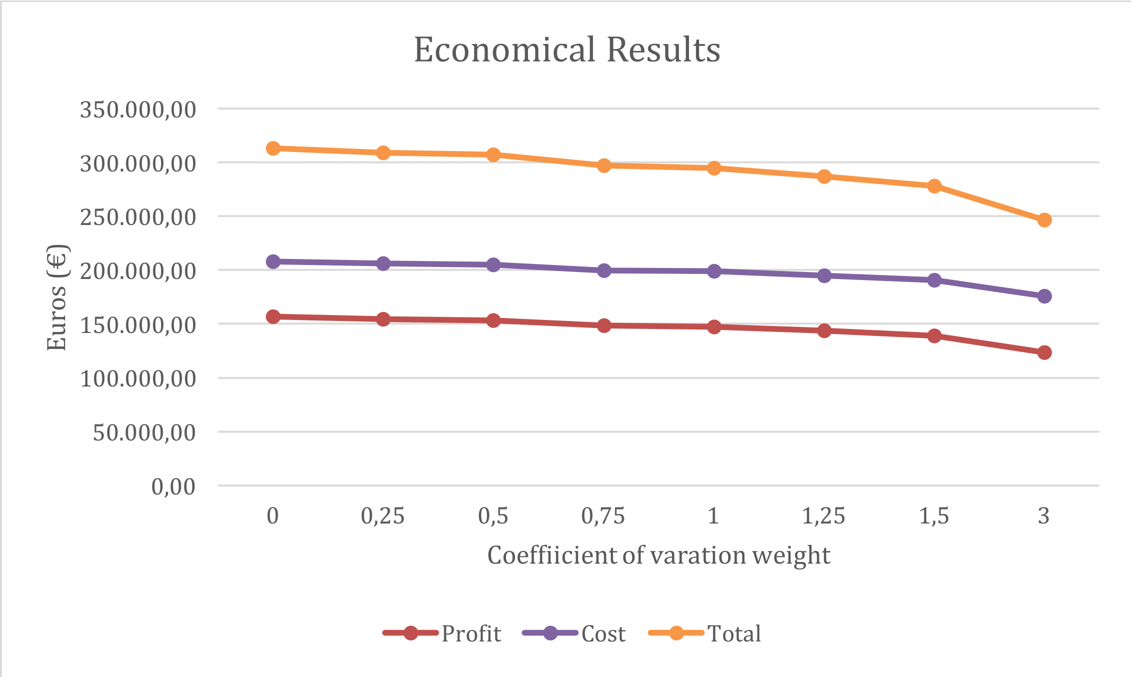


Figure 21: Economical results after an optimization

Source 21: Own

It is also obtained an optimal scheduling of number of pigs delivered to the slaughterhouse, which is shown below.

WEEK/ COEF.	*w8*	*w9*	*w10*	*w11*	*w12*	*w13*	*w14*	*w15*	*w16*	*w17*
0	0	0	0	0	0	0	0	0	0	1000
0.25	0	0	0	0	0	0	0	0	0	1000
0.5	0	0	0	0	0	0	0	0	100	900
0.75	0	0	0	0	0	0	0	100	0	900
1	0	0	0	0	0	0	100	0	100	800
1.25	0	0	0	0	0	100	0	180	0	720
1.5	0	0	0	0	100	100	0	0	100	700
3	100	0	0	100	0	100	0	100	0	600

Table 6: Optimal scheduling delivered to the slaughterhouse

According to previous results, a farmer would prefer a homogenous batch, since it is easier to manage, farmers have more control over the marketing of animals and also they can reduce the variation in marketing time window. Moreover, having a homogeneous batch means that the slaughtering and subsequent meat processing are conducted more efficiently. Consequently, different coefficients of homogeneity were evaluated, meaning from no variation of the weight of the pig to three times the regular variation. The economical results in *Table 5* show that more heterogeneous the batch is, the less profit the farmer have. The difference between no variation and three times the regular variation means a loss of more than 30.000 euros. Therefore, the coefficient of homogeneity is an important factor which must be taken into account.

The results obtained for the basic situation indicates that during the 17th week of the marketing period the whole batch has been sent to the slaughterhouse. The total amount of the trucks used are six. Four trucks were needed to deliver 80% of pigs to the slaughterhouse on the last week (Week 17 of fattening) and the other two trucks at the week 23 and 25, delivering 10% of pigs respectively (Week 14 and 16 of fattening). The heaviest pigs in the batch are always the first ones to be sent to the slaughterhouse, which this can be realistic for the farmer. This mentioned results can be seen in *Table 6*.

The results in *Table 6* shows the optimal scheduling number of the pigs which have to be delivered to the slaughterhouse depending on the homogeneity of the batch. It can be seen that the more heterogeneous the batch is, the more dispersed the the shipments to the slaughterhouse are. For instance, when the batch is homogenous, the whole batch is sold to the slaughterhouse at a time. At the 17th week, 1,000 pigs are sent to the slaughterhouse which represents the entire batch. However, the shipments of pigs are sparser and more periodic when the batch is more heterogeneous, as it can also be seen in this table. Having a look at the variation of weight when it is three times the regular one (i.e. the model considers extremely heterogeneous growth), the 10% of animals are sent in week 8, 11, 13, 15 of fattening period and the rest (60% of the batch) are sent the last week (Week 17 of fattening).

Another important factor that must be considered when sending pigs to the slaughterhouse is the price-grid system (*Table 1*). For this reason, the graph presented in *Table 6* is the optimal scheduling to the slaughterhouse, since this economical factor has been taken into consideration.

Two following scenarios can be found on a day-to-day basis. The first scenario is that there is no bonus nor penalties on age or weight, in other words, payment only by live weight. Therefore, this means that the overall weight of the animal is boosted, regardless of the amount of fat it contains. The second scenario is one that takes into account the model since it is the most usual case nowadays. This second scenario is one that the farmer has to foresee the production of fat of the animal to not have many losses when selling the animal. As a result, it can be seen in *Table 6* when pigs are more heterogeneous, for instance, with coefficient 3, a 10% of pigs in that batch are sold earlier (Week 8 of fattening). This is the optimal time that the animal would generate more benefit. This 10% of pigs, if they had stayed in the farm for more weeks, their benefit would be lower because from that moment, they would create fat and this would suppose a penalty by the slaughterhouse when selling them. To analyse the impact that price-grid systems have, higher bonuses were applied to the less heavy pigs (i.e. pigs weighing from 50 to 70 kg).

The obtained results have demonstrated that the price-grid system also can have an impact on the optimal marketing policy. Therefore, farmers have to know the pricing-grid system in advance in order to value the economic factors and thus be able to maximize the benefits as well as satisfy better the needs of their customers.

4. CONCLUSIONS

This project describes the development of a decision support system, which is related to the optimal delivery of fattened pigs to the slaughterhouse, and its integration in a cloud platform has served to offer the benefits of practical optimization models as a service through a client application.

The integration into the decision making process of optimisation models gives additional information to the decision maker and allows him to make better decisions being beneficial for the company. In addition, the capability of sensitivity analysis based on the optimization model has an important and critical impact in the practical decision-making, since ignoring the relevance of some factors may mean losses for the pig producers. The results presented have demonstrated that several factors such as the batch homogeneity or price-grid system have an enormous impact on the optimal marketing policy of fattening farms with important consequences in vertically integrated companies. Pig producers must be aware of the existence of such factors correlated positively with the increase of their benefits and value fairly the impact.

On the other hand, the client application has been successfully developed, and enables the user to interact with the mathematical model, either executing an optimization or performing a sensibility analysis. The consequences are that the farmers are able to control their farm more tightly through the displayed information through the interface. This interface has been designed to be used intuitively for any person operating an electronic device and without having an excellent computer science skill.

Regarding the architecture of this project, it has been an advantage for the developer to use a private cloud and a micro-service, since they have provided elasticity in the executions. This micro-service is responsible for the management of the executions on the cloud. Moreover, replacing OPL as a modelling language and starting to use Pyomo that allows to use Python as a programming language has reduced complexity to the automation of the application and left behind that rudimentary process, complex and difficult to maintain out of academic purposes.

Finally, I value positively that these both parts of the project have had successful results, since I have managed to do what was proposed at the beginning of the project. In particular, the client is now able to interact with the model through the designed client application. The client can also analyse the results, explore alternatives, draw conclusions and then, make better decisions. However, this is just a module of a huge project and thus, it will be subject to new future changes. This project has to be reviewed periodically in order to modify or add new parameters, to extent new functionalities or cover new needs requested by users in the future.

5. FUTURE WORK

This project is expected to grow over time having improvements and updates. Therefore, below there are four extensions described that may be interesting to be applied in the future.

1) Integration of this platform with 4.0 farms that have built-in sensors to adjust the growth curves or consumption much better. This way, the user would no longer enter any index or variable through the client application, since it would be done automatically from the sensors. The sensor data would be transferred to the server, the server would make the calculations and the user would receive the information. Once a week, the user would have available information relevant to select the worst animals to be sent to the slaughterhouse.

2) Integration of a market price forecasting system. If next week sale price would be predicted, it would imply that the delivery time of the pigs could be known more precisely in advance. Meaning that sending the animals to the slaughterhouse next week or waiting better an extra week because the prediction says that in two weeks the price will rise.

3) Working with more than one slaughterhouse. Therefore, the farmer could see which slaughterhouse is most likely to send the fattened pigs, in order to have a greater income.

4) Adapting this model from the slaughterhouse point of view to build a grid of prices according to consumer wishes and likes.

6. GREETINGS

I would like to thank my both tutor of this thesis: Jordi Mateo and Lluís Miquel Pla for giving me the opportunity to participate on this project and also for guiding me throughout this year. Besides this, thanks for your patience attending all the doubts I had and for the confidence deposited on me from the beginning. At the same time, I would like also thank all the members of this whole project, Esteve Nadal, Adela Pagès and the other members of the project for their help and support whenever I needed it.

A sincerely thanks is dedicated to my family and all my friends for their support. I am so grateful since they have been always by my side either on the good and bad moments throughout the thesis.

7. BIBLIOGRAPHY

Rodríguez-Sanchez S.V., Pla-Aragones L.M & De Castro R. (2018). Insights to optimise marketing decisions on pig-grower farms. *CSIRO Publishing – Animal Production Science*, doi: 10.1071/AN17360

[1] Magapor (2018). The swine sector in Spain and its evolution. *Retrieved from: https://www.magapor.com/el-sector-porcino-espanol/*

[2] https://ec.europa.eu/eurostat/statistics-explained/index.php/Glossary:Carcass_weight

[3]https://publications.europa.eu/en/publication-detail/-/publication/9716803a-8887-4956-9877-629031ec7723/language-en

[4] https://stormy.udl.cat

[5] https://www.ibm.com/analytics/cplex-optimizer

[6] https://www.jetbrains.com/

[7] http://www.pyomo.org/

[8] https://www.primefaces.org/primeng/#/

[9] https://stackoverflow.com

[10] https://yogaysenderismo.com/nuestras-actividades/icono-web-formulario-2/

[11] https://www.onlinewebfonts.com/icon/263989

[12] https://es.pngtree.com/freepng/cloud-data-vector_3353008.html

[13]https://www.3gi.es/index.php/3gi-servicios/3gi-portfolio/18-3gi-analisis-informatico

[14] https://www.edp.tech/

[15] https://devcode.la/blog/frontend-y-backend/

[16] https://sistemas.com/usuario.php

ANNEX 1: Growth and feed-intake curves by week

	Weight (kg)		Intake (kg)	
Weeks	Mean	s.d	Mean	s.d
10	29,7	3,9	5,1	5,5
11	33,4	4,6	12,1	8,5
12	37,8	5,4	20,5	12,1
13	42,6	6,3	30,2	15,9
14	47,9	7,4	41,3	19,7
15	53,5	8,4	53,4	23,6
16	59,3	9,5	66,4	27,5
17	65,3	10,6	80,3	31,4
18	71,3	11,8	94,9	35,3
19	77,4	12,9	110,1	39,2
20	83,4	14	125,7	43,2
21	89,2	15,2	141,6	47,1
22	94,8	16,3	157,6	51
23	100	17,5	173,7	54,9
24	104,8	18,7	189,6	58,9
25	109,1	19,8	205,3	62,8
26	112,8	21	220,6	66,7

ANNEX 2: SEUROP class distribution based on liveweight and carcass weight

S	E	U	R	O	P	Liveweight (kg)	Carcass weight (kg)
0,57	0,28	0,12	0,03	0,00	0,00	50,00	39,40
0,55	0,27	0,14	0,04	0,00	0,00	55,00	43,40
0,53	0,26	0,15	0,05	0,01	0,00	60,00	47,40
0,50	0,28	0,16	0,05	0,01	0,00	65,00	51,40
0,49	0,27	0,17	0,05	0,01	0,00	70,00	55,50
0,49	0,28	0,17	0,06	0,01	0,00	75,00	59,50
0,45	0,27	0,18	0,08	0,01	0,00	80,00	63,60
0,44	0,26	0,19	0,09	0,03	0,01	85,00	67,60
0,43	0,25	0,18	0,09	0,03	0,01	90,00	71,70
0,41	0,24	0,19	0,10	0,04	0,01	95,00	75,80
0,40	0,23	0,19	0,11	0,05	0,02	100,00	79,90
0,39	0,23	0,19	0,12	0,05	0,02	105,00	84,00
0,38	0,22	0,19	0,12	0,06	0,03	110,00	88, 10
0,38	0,21	0,19	0,13	0,07	0,04	115,00	92,30